Uroburos – Deeper travel into kernel protection mitigation

By MN

Published: 2017-05-11 · Archived: 2026-04-05 14:34:45 UTC

First of all, we would like to send regards and thanks to the people being active on the kernelmode.info forum , in particular, R136a1 and EP_X0FF. They provided a proficient analysis of the Driver Signature Enforcement bypass which enriches the overall understanding of the case.

Introduction

The following analysis article is closely linked to G DATA's Red Paper about Uroburos, published on Friday, February 28th. The paper can be downloaded here: https://secure.gd/dl-en-rp-Uroburos
For fellow researchers, we provide the hash of the sample used for this subsequent article:
SHA256: 33460a8f849550267910b7893f0867afe55a5a24452d538f796d9674e629acc4
This file is a 64-bit driver, compiled in 2011.

Kernel Patch Protection

Definition

The majority of rootkits mainly use kernel modification or kernel patching to hide their activities and modify the behavior of the infected system. To protect the Windows operating system, Microsoft added a new technology to its 64-bit Windows editions. The Kernel Patch Protection technology (aka PatchGuard) checks the integrity of the Windows kernel to make sure that no critical parts are modified. In case a harmful modification of the kernel is detected, the KeBugCheckEx() function is executed, called with an argument with the value 0x109 (CRITICAL_STRUCTURE_CORRUPTION) as bug code. The result is a shutdown of the system with a blue screen.

Microsoft describes that the Kernel Patch Protection technology prevents the following modifications:

- modify system services tables, for example, by hooking the KeServiceDescriptor table
- modify the Interrupt Descriptor Table (IDT)
- modify the Global Descriptor Table (GDT)
- use kernel stacks that are not allocated by the kernel
- patch any part of the kernel

Uroburos mitigation

Uroburos' developers used the same inline hooks, explained in our previous Red Paper, to bypass Kernel Patch Protection. The attacker's goal is to hook the KeBugCheckEx() function to avoid handling the bug code 0x109. The screenshot below shows the code snippet in which the address of KeBugCheckEx() is stored in qword_787D8

In case you wish to get more information concerning the technique used by the developers, we suggest the following link: http://www.codeproject.com/Articles/28318/Bypassing-PatchGuard

Driver Signature Enforcement

Definition

Rootkits are usually drivers which used to work in kernel space. To avoid this kind of malware, Microsoft created a Driver Signing Policy for its 64-bit versions of Windows Vista and later versions. To load a driver, the .sys file must be signed by a legitimate publisher.
Developers may disable the Driver Signature Enforcement process during the development phase of a driver, which means a developer does not have to sign each compiled driver version during development phase. In this case, the desktop of the machine is changed and the following message appears in the bottom right corner: “Test Mode”. The flag with which the current status of the policy is stored is called g_CiEnabled. The value of g_CiEnabled is set during the Windows boot phase, and considered “static" during runtime. This means, Windows assumes the value is set correctly and does not change during runtime.

Uroburos mitigation

Uroburos’ developers used new techniques to disable the Driver Signature Enforcement. In our case, they used a vulnerability in a legitimately signed driver to disable the policy! During the installation of Uroburos, the Oracle VirtualBox driver (version 1.6.2) is installed on the targeted system. This driver (VBoxDrv.sys) is signed:

Uroburos’ developers used new techniques to disable the Driver Signature Enforcement. In our case, they used a vulnerability in a legitimately signed driver to disable the policy! During the installation of Uroburos, the Oracle VirtualBox driver (version 1.6.2) is installed on the targeted system. This driver (VBoxDrv.sys) is signed:

Conclusion

Previously, we have claimed that Uroburos is a highly complex and very sophisticated malware, programmed by skilled people. This assumption is corroborated once more by the aforementioned analysis of Uroburos’ installation technique.
The developers had to deal with Microsoft Windows security enforcement. They had to find ways to bypass the Kernel Patch Protection technology and also the Driver Signature Enforcement. The technique used to bypass the Kernel Patch Protection has been documented on the Internet and therefore is not absolutely new.

But, concerning the Driver Signature Enforcement, this is the first time that we see a malware using a vulnerability in a legitimately signed driver to disable the policy!
This example shows the limitation of the signature process. Generally, the signature expiration date is set to happen several years after its creation date. In case any vulnerability is found, a patch is provided, but the old binary is still available and valid, except in case the certificate is revoked by the author/signer and set onto the CRL, the certificate revocation list.

But, revoking a signature is only the first step in the protection process, because each and every system that needs to check a signature needs to have access to an up to date CRL.
And even in case the system has an updated CRL, the Uroburos authors are certainly thought to be skilled enough to manipulate the verification process the operation system is using without alerting the user.
So, it is the first time that we see those two techniques to bypass Windows' protection mechanisms in the wild. We expect that they will be used by more malware in the future, of course.

In case someone from the audience notices an infection caused by the Uroburos rootkit and needs help, would like to receive further technical information or would like to contribute any information about this case, please feel free to contact us by email using the following mailbox: intelligence@remove-this.gdata.de

---

G Data's Uroburos analysis red paper:
https://secure.gd/dl-en-rp-Uroburos
